Python Tutorial

Projekt Collaborative Writing

Hochschule Kaiserslautern

Inhaltsverzeichnis

Kapitel 1

Objekte und Klassen

Python ist wie Java eine Objektorientierte Programmiersprache. Das bedeutet, dass in Python fast alles aus Objekten und Klassen besteht. Klassen sind Vorlagen, aus denen Objekte generiert werden können. Dabei enthält die Klasse je nach Verwendungszweck Variablen und Funktionen. Ein klassisches Beispiel wäre die Klasse „Schüler“. Jeder Schüler hat einen Namen, aber nicht jeder Schüler hat den gleichen Namen. Um dies Anhand eines Python Programms zu verdeutlichen wird ein neues Programm erstellt

```
class Schueler:
        name = "Name einfuegen"
```

Mithilfe der oben erstellten Klasse kann man nun verschiedene Objekte erstellen, welche die Variablen und Funktionen der Klasse enthalten.

```
class Schueler:
name = "Name einfuegen"
schueler1 = Schueler()
print(schueler1.name)
```

Ausgabe des Programmcodes:

```
        Name einfuegen
```

Die Variablen der Objekte sind zunächst gleich mit denen der Klasse, aus der diese erstellt wurden. Allerdings sind die Variablen des erstellten Objekts unabhängig von denen der Klasse. Das bedeutet, dass man diese auch unabhängig für jedes einzelne Objekt ändern kann. Die im obigen Beispiel verwendete Klasse ist in realen Anwendungen nicht verwendbar, da die Attribute des Objekts von Anfang an festgelegt wurden. Um einen dynamischen Ansatz zu nutzen, sollte man wie im nächsten Beispiel vorgehen.

```
class Schueler:
        def __init__(self, name):
                self.name = name

schueler1 = Schueler("Patrick")
schueler2 = Schueler("Sebastian")
print(schueler1.name)
print(schueler2.name)
```

Ausgabe des Programmcodes:

```
Patrick
Sebastian
```

Jede Klasse hat eine init() Funktion, die immer ausgeführt wird, wenn die Klasse initiiert und ausgeführt wird. Diese Funktion wird verwendet um den Variablen des Objektes einen Wert zu geben. Hierbei ist der self Parameter notwendig um die Klasse selbst zu referenzieren und um auf die Variablen der Klasse zugreifen zu können. Dieser Parameter ist also notwendig, muss aber nicht self genannt werden, sondern kann einen beliebigen Namen haben. Er wird als erster Parameter bei jeder Funktion angegeben.

```
class Schueler:
        def __init__(self, name):
                        self.name = name

        def getName(self):
                print(self.name)

schueler1 = Schueler("Patrick")
schueler2 = Schueler("Sebastian")
schueler1.getName()
schueler2.getname()
```

Ausgabe des Programmcodes:

```
Patrick
Sebastian
```

Objekte enthalten die aus den Klassen übernommenen Funktionen. Diese Funktionen gehören jetzt dem Objekt und werden Methoden genannt. Um Parameter zu modifizieren oder zu löschen, können einfach die Befehle ob-

jektname.parameter = neuer Wert (modifizieren) und del objektname.parameter (löschen) verwendet werden.

```
class Schueler:
def __init__(self, name):
self.name = name

def getName(self):
print(self.name)

schueler1 = Schueler("Patrick")
schueler2 = Schueler("Sebastian")
schueler1.name = "Lukas"
schueler1.getName()
schueler2.getname()
del schueler2.name
```

Ausgabe des Programmcodes:

```
Lukas
Sebastian
```

Ein weiterer wichtiger Aspekt der Klassen in Python ist die Vererbung und Ergänzung einer Klasse.

```
class Schueler:
        def __init__(self, name):
        self.name = name

        def getName(self):
        print(self.name)

class Hochschule:
        def __init__(self, name, matrikelnummer):
        Schueler..__init__(self, name)
        self.matrikelnummer = matrikelnummer

        def getName(self):
        print(Schueler.getName(self) + ", " + self.matrikelnummer)

schueler1 = Hochschule("Patrik","1234")
schueler2 = Hochschule("Sebastian", "1235")
schueler1.getName()
schueler2.getname()
```

Ausgabe des Programmcodes:

```
Patrick, 1234
Sebastian, 1235
```

In diesem Beispiel wurde die Klasse Schüler durch die Klasse Hochschule erweitert. Dies geschieht durch das Hinzufügen eines weiteren Attributes (bspw. Matrikelnummer). Dieses Attribut hat dann die Funktion getName abgeändert. Die Klasse, von der geerbt wird, kann man entweder mit dem Klassennamen (Schueler) oder mit super referenzieren. Trotz der Änderung der Funktion getName in der Klasse Hochschule, behält die Klasse Schüler ihre Funktion.

Zum Schluss bleibt zu erwähnen, dass auch komplette Objekte löschbar sind. Dies ist mit dem Befehl del objektname möglich. Hierbei wird eine Fehlermeldung ausgegeben, da versucht wird, auf das gelöschte Objekt zuzugreifen.

```
class Schueler:
def __init__(self, name):
self.name = name

def getName(self):
print(self.name)

schueler1 = Schueler("Patrick")
schueler2 = Schueler("Sebastian")
schueler1.name = "Lukas"
schueler1.getName()
schueler2.getname()
del schueler2.name
del schueler2
schueler2.getName()
```

Fehlermeldung beim Ausführen des Programmcodes:

```
Traceback (most recent call last):
Lukas
        File"C:/users/Patrick/Desktop/python/objekte
        _undklassen.py", line 15, in <module>
Sebastian
        schueler2.getName()
NameError: name 'schueler2' is not defined
```

1.0.1 Übungen

1.) Ergänzen Sie die fehlenden Code Fragmente damit die vorgegebene Ausgabe entsteht

```
class Person:
        def __init__(self,                              );









        person1 = Person("Peter", "Petersen")
        person2 = Person("Axel", "Schweiss")
        person1.getName()
        person2.getName()
```

Ausgabe Programmcode:

```
Peter Petersen
Axel Schweiss
```

2. Ergänzen Sie die folgende Klasse (mit Vererbung) um die unten stehende Ausgabe zu erhalten.

```
class Auto:
        def __init__(self, hersteller, farbe):
        self.hersteller = hersteller
        self.farbe = farbe

        def getAuto(self):
                return self.hersteller + " " + self.farbe
```

Ausgabe des Programmcodes:

```
VW rot Polo
```

Literaturverzeichnis

[Pil09] PILGRIM, MARK: *Dive Into Python 3*. Apress, 2009.